# AN ALMOST LINEAR-TIME ALGORITHM FOR GRAPH REALIZATION

Robert E. Bixby[1]

Donald K. Wagner[2]

Technical Report 85-2, March 1985.

[1]Mathematical Sciences Department, Rice University, Houston, Texas 77251. Research partially supported by NSF Grant MCS81-052327 and AFOSR Grant AFOSR82-0004 at Northwestern University, and by the Alexander von Humboldt Foundation and the Institut für Ökonometrie und Operations Research der Universität Bonn, Federal Republic of Germany, while the author was on leave from Rice University.

[2] School of Industrial Engineering, Purdue University, West Lafayette, Indiana 47907. Research partially supported by NSF Grant MCS81-05327 and AFOSR Grant AFOSR82-0004 at Northwestern University, and by NSF Grant ECS-8307796 at Purdue University.

# Report Documentation Page

*Form Approved*
*OMB No. 0704-0188*

| | | |
|---|---|---|
| 1. REPORT DATE **MAR 1985** | 2. REPORT TYPE | 3. DATES COVERED **00-00-1985 to 00-00-1985** |
| 4. TITLE AND SUBTITLE **An Almost Linear-Time Algorithm for Graph Realization** | | 5a. CONTRACT NUMBER |
| | | 5b. GRANT NUMBER |
| | | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | | 5d. PROJECT NUMBER |
| | | 5e. TASK NUMBER |
| | | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Computational and Applied Mathematics Department ,Rice University,6100 Main Street MS 134,Houston,TX,77005-1892** | | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | | 10. SPONSOR/MONITOR’S ACRONYM(S) |
| | | 11. SPONSOR/MONITOR’S REPORT NUMBER(S) |
| 12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited** | | |
| 13. SUPPLEMENTARY NOTES | | |
| 14. ABSTRACT | | |
| 15. SUBJECT TERMS | | |

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT **unclassified** | b. ABSTRACT **unclassified** | c. THIS PAGE **unclassified** | | **42** | |

**Standard Form 298 (Rev. 8-98)**
Prescribed by ANSI Std Z39-18

# An Almost Linear-Time Algorithm for Graph Realization

Robert E. Bixby and Donald K. Wagner

## Abstract

Given a $\{0,1\}$-matrix $M$, the graph-realization problem for $M$ is to find a tree such that the columns of $M$ are incidence vectors of paths in $T$, or to show that no such $T$ exists. An algorithm is presented for this problem the time complexity of which is *very nearly* linear in the number of ones in $M$.

## 1. INTRODUCTION

The purpose of this paper is to describe an efficient algorithm for graph realization.[1] Our motivation is the applicability of such an algorithm to finding network structure in linear programs (see [2,11]). Graph realization also provides a generalization of planarity testing.

Fujishige [7] has independently[2] given an algorithm with essentially the same time bound as that given here. However, many of the details in [7] are left to the reader making it difficult to understand and verify the algorithm. The algorithm in [7] and the one given here are similar in that they both are based on an exponential-time procedure proposed in [13]. Fujishige makes this procedure polynomial by employing the PQ-tree data structure of [4]. The algorithm given here does not use any such special data structure, using instead a graph-decomposition.

There are several equivalent definitions of the ***graph-realization problem***. The simplest is: Given a $\{0,1\}$-matrix $M$, determine if a tree $T$ exists such that the columns of $M$ are incidence vectors of paths of $T$, and if so find such a tree. (Note: It is not necessary that all paths of $T$ occur as columns in $M$, only that all columns of $M$ occur as paths in $T$.) If $T$ exists, $M$ is called *graphic*.

The first polynomial-time algorithm for graph realization was given by Tutte [19]. Other polynomial algorithms are given in [2,5,7,9,11,14,16,18,20]. Our algorithm and the algorithm in [7] have the property that for an $M$ that is not "entirely graphic," a maximal subset of "graphic columns" is produced. This property is important in the application to linear programming.

The remainder of the paper is arranged as follows. Section 2 supplies several definitions. In section 3, Löfgren's procedure is given along with an informal discussion of how to make it polynomial. In sections 4 and 5 the main subroutines are developed in detail -- TYPING, HYPOPATH and UPDATE. In section 6 the main algorithm GRAPH is stated. Section 7 contains the derivation of the complexity of the algorithm. Finally, an example is given in the Appendix.

[1]The paper is based on the Ph.D. dissertation [22] of the second author. An outline of the main algorithm has also recently been given in [1].

[2]Preliminary results of the current paper were presented at the April 30-May 2, 1979 TIMS/ORSA national meeting.

## 2. DEFINITIONS

We assume a basic knowledge of graph theory. See [3] for an introduction.

Let $G = (N, E)$ be a graph with node-set $N$ and edge-set $E$. An edge with identical ends is a *loop,* and two non-loop edges with the same ends are *parallel.* Let $A$ be a set with $A \subseteq E$. The *edge-induced subgraph* $G[A]$ is the subgraph with edge-set $A$ and node-set consisting of those nodes meeting $A$. $G \backslash A$ is the spanning subgraph obtained by deleting the edges of $A$. A path is a "simple path" (one without repeated nodes), and a cycle is a "simple closed path." We identify paths and cycles with their underlying edge-sets. A *polygon* is a connected graph the edge-set of which is a cycle with at least three edges. A connected, loopless graph on two nodes is a *bond.*

Let $T = (N, A)$ be a directed graph (digraph) with arc-set $A = A(T)$. $T$ is an *arborescence* if its underlying graph is a tree, every node except one has indegree one, and the one special node, denoted $root(T)$, has indegree zero. Let $u$ and $v$ be the tail and head, respectively, of some arc of $T$. Then $u$ is the *parent* of $v$ and $v$ a *child* of $u$. Note that every node $u$, except $root(T)$, has a unique parent, denoted here by $p(u)$.

Let $G = (N, E)$ be a connected graph with spanning tree $T$. Then for each edge $e \in E - T$ the graph $G[T \bigcup \{e\}]$ contains a unique cycle, denoted $C(T, e)$, the *fundamental cycle* of $T$ determined by $e$. Define a $\{0,1\}$-matrix $M$ as follows. Index the columns of $M$ on $E - T$, the rows of $M$ on $T$ and define $m_{j,k}$, the $(j, k)$-entry of $M$, by

$$m_{j,k} = \begin{cases} 1 & \text{if } j \in C(T, k) \\ 0 & \textit{otherwise} \ . \end{cases}$$

$M$ is a *fundamental matrix* of $G$ with respect to $T$.

Given a $\{0,1\}$-matrix $M$, the *graph-realization problem* is to determine a connected graph $G$, if one exists, such that $M$ is a fundamental matrix for $G$. $M$ is then said to be *graphic* and $G$ is a *realization* of $M$. This definition is equivalent to the one in the Introduction. For any $\{0,1\}$-matrix $M$, graphic or not, the set of row indices corresponding to nonzeros in column $k$ together with the column index $k$ will be denoted by $C_k$ and be referred to as a "fundamental cycle" of $M$.

Let $G = (N, E)$ be a connected graph, and let $\{E_1, E_2\}$ be a partition of $E$. Then, for $k > 0$, $\{E_1, E_2\}$ is a ***k-separation*** of $G$ if

$$|E_1| \geq k \leq |E_2|$$

$$|N(G[E_1]) \bigcap N(G[E_2])| = k$$

For $n$ a positive integer, $G$ is ***n-connected*** if it has no $k$-separation for $k < n$. Graphs having a 1-separation are ***separable;*** 2-connected graphs are ***nonseparable.***

Let $\{E_1, E_2\}$ be a 2-separation of a nonseparable graph $G$. Let $N(G[E_1]) \bigcap N(G[E_2]) = \{u, v\}$. Define $G'$ to be the graph obtained by interchanging in $G[E_1]$ the incidences of the nodes $u$ and $v$. Then $G'$ is said to be obtained from $G$ by ***reversing*** $G[E_1]$. In general, $G''$ is ***2-isomorphic*** to $G$ if $G''$ is obtainable from $G$ by a sequence of subgraph reversals.

A ***parallel class*** of a graph $G$ is a maximal set of edges $B$ such that $G[B]$ is a bond. "The" ***simplification*** of $G$ is the graph obtained by deleting all loops and all but one edge in each parallel class. If $G$ is not a polygon or bond and its simplification is 3-connected, then it is ***prime.***

The following important theorem is due to Whitney [24].

(2.1) **Theorem.** Let $G$ and $G'$ be nonseparable graphs on the same edge-set. Then $G$ and $G'$ are 2-isomorphic if and only if they have the same cycles.

●

For a short proof of Theorem 2.1, see [17] or [23].

## 3. AN OUTLINE OF THE ALGORITHM

Let $M$ be an $r \times c$ $\{0,1\}$-matrix. Assume $M$ has no row or column consisting of zeros only, and let $R$ and $C$ be its sets of row and column indices, respectively. Define a graph $H$ with node-set $R \bigcup C$ and an edge for each nonzero of $M$. The submatrices corresponding to the connected components of $H$ are called the ***blocks*** of $M$, and $M$ is said to be ***nonseparable*** if it has just one block. The following result is well known.

(3.1) **Theorem.** $M$ is graphic if and only if each block of $M$ is graphic; moreover, if $M$ is nonseparable and graphic with realization $G$, then $G$ is nonseparable.

●

Linear-time algorithms to find the blocks of $M$, linear in the number of nonzeros, are easy to describe (e.g., see [2]).

Let $M_k$ denote the matrix made up of the first $k$ columns of $M$, where rows consisting entirely of zeros have been deleted. $M$ is called ***totally nonseparable*** if $M_k$ is nonseparable for $1 \leq k \leq c$. For any nonseparable matrix it is easy to compute a permutation of the columns such that the resulting matrix is totally nonseparable. For example, virtually any algorithm for computing the components of $H$, above, will so order the columns.

In view of the above remarks we assume henceforth that $M$ is totally nonseparable. We also assume, for purely technical reasons, that the first column of $M$ is a singleton (see the first paragraph of section 4).

Where $C_k$ is the fundamental cycle of column $k$ of $M$, define $P_k = C_k \bigcap \left\{ \bigcup_{j<k} C_j \right\}$. A set of edges $P$ of a graph $G$ is a ***hypopath*** of $G$ if $P$ is a path in some graph 2-isomorphic to $G$. The following statement, provable directly from Theorem 2.1, is Löfgren's "subrearrangement theorem" [13].

(3.2) **Theorem.** Let $M$ be a totally nonseparable $\{0,1\}$-matrix with $c$ columns. Assume for some $1 \leq k < c$ that $M_k$ is graphic with realization $G_k$. Then $M_{k+1}$ is graphic if and only if $P_{k+1}$ is a hypopath of $G_k$.

●

Based on Theorem 3.2, Löfgren suggested the following procedure for testing whether $M$ is graphic. Clearly $M_1$ is graphic. Suppose there exists a graph $G_k$ that realizes $M_k$. Further, suppose $P_{k+1}$ is a hypopath of $G_k$. Then there exists a graph $G_k'$, 2-isomorphic to $G_k$ such that $P_{k+1}$ is a path in $G_k'$. Add the edges of $C_{k+1} - P_{k+1}$ to $G_k'$ so that they form a path between the ends of $P_{k+1}$ but are not incident to any other nodes of $G_k'$. It is straightforward to verify that the resulting graph $G_{k+1}$ is a realization of $M_{k+1}$. If the above procedure breaks down at any point, it follows that $M$ is not graphic.

To implement Löfgren's idea requires a polynomial-time method for constructing $G_k'$ from $G_k$. A natural approach is to invoke some representation of $G_k$ that "displays" all graphs 2-isomorphic to $G_k$. For this representation we use a graph decomposition, called here t-decomposition. This notion of decomposition is a small variation on a theory developed by Tutte [21, Chapter 11]. The data structures needed to represent a t-decomposition are elementary.

The remainder of the section is devoted to describing an example decomposition and illustrating pictorially how it can be used to implement the above procedure.

A careful definition of t-decomposition will be given in the next section. For now we define a t-decomposition as an arborescence whose nodes are graphs. Two nodes, that is graphs, are adjacent if they have an edge in common, called a "marker" edge. It is assumed that each graph is a bond, polygon or prime. An example t-decomposition is shown in Figure 1a. The arrows between graphs indicate the pairings of the marker edges. The root is $G_1$. The marker edges themselves are indicated by oriented dashed lines.

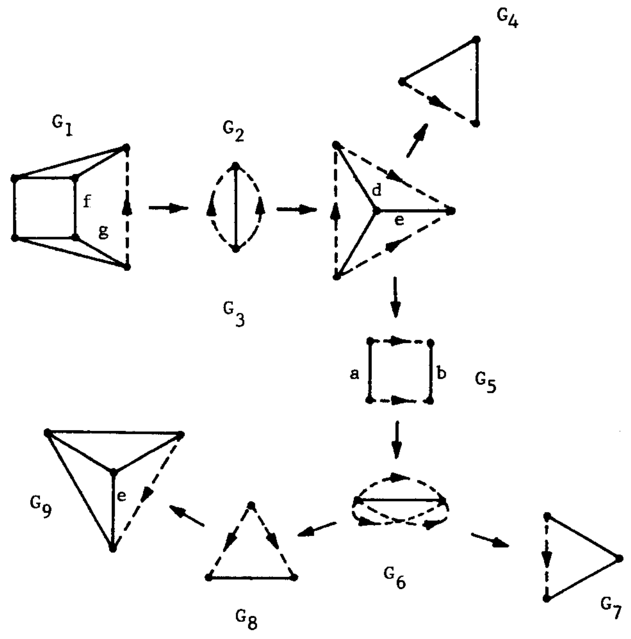

**Figure 1a:** A t-decomposition

We associate with each t-decomposition a "merged" graph obtained by identifying the common marker edges and then erasing them. The merged graph for the graph in Figure 1a graph is shown in Figure 1b. The orientations on the marker edges specify the orientations in which they are identified.

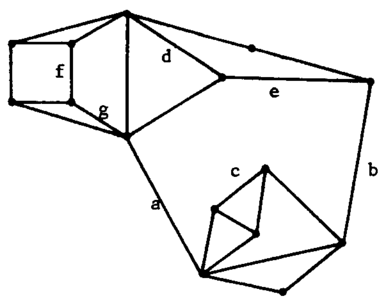

**Figure 1b:** The merged graph for Figure 1a

Let $G$ be the graph in Figure 1b and let $P = \{a,b,c,d,e,f,g\}$. $P$ plays the role of $P_{k+1}$ in the Löfgren procedure. Each edge of $P$ lies in one of the graphs of the t-decomposition $D$ in Figure 1a. Consider the t-decomposition $D'$ in Figure 2a.

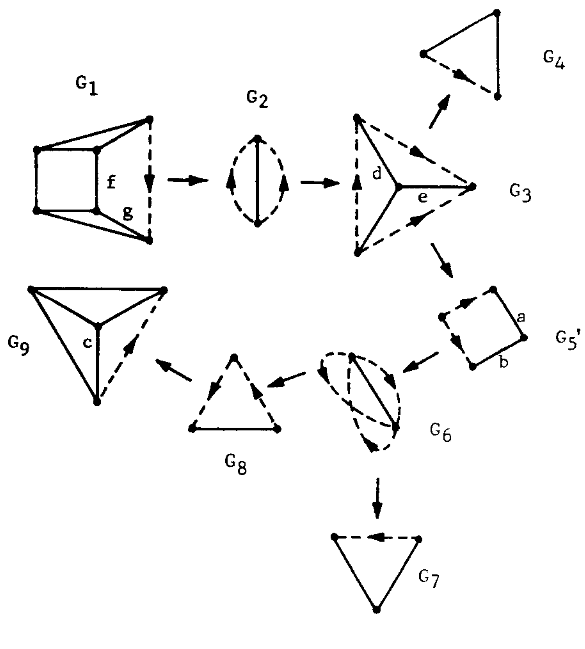

**Figure 2a:** Altered t-decomposition

$D'$ is obtained from $D$ by "reorienting" $G_1$ with respect to $G_2$ (that is, by reorientation of its marker edge), "relinking" the polygon $G_5$ (that is, by reordering its edges) and reorienting $G_9$ with respect to $G_8$. The result is that the merged graph $G'$ of $D'$ is as given in Figure 2b and has the property that $P$ is a path. Thus, $P$ is a hypopath of $G$. An algorithm HYPOPATH for carrying out the above kinds of operations -- marker reorientations and polygon relinkings -- is the subject of section 4.

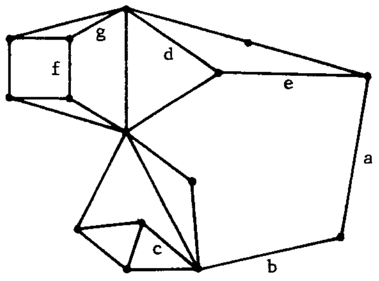

**Figure 2b:** The merged graph for Figure 2a

The final step in an efficient implementation using t-decompositions is to give an update procedure. (A polynomial-time but inefficient alternative would be to recompute the decomposition from scratch [10].) Thus, suppose $h$ is an edge not in $G'$, and that $h$ is added to $G'$ forming a cycle with $P$. We must construct the t-decomposition for the new graph. It is given in Figure 2c. It was obtained by merging $G_1$, $G_2$, $G_3$, $G_5'$, $G_6$, $G_8$ and $G_9$, in the process of which the piece $G_5''$ was "squeezed" off. Section 5 describes the general procedure for this update, the algorithm UPDATE.

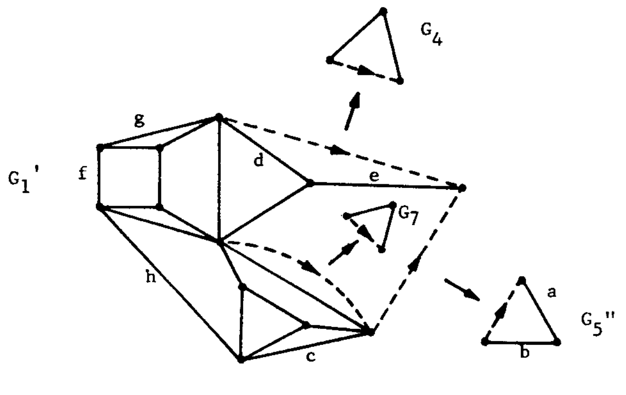

**Figure 2c:** Updated t-decomposition

## 4. THE HYPOPATH PROBLEM

We begin by describing a graph decomposition. Let $D$ be a finite collection of nonseparable graphs, and let $T$ be a digraph having node-set $D$ and two nodes joined by an arc if these nodes have some common edge. The collection $D$ is a *decomposition* if $T$ is an arborescence and any two members of $D$ have at most one edge and no nodes in common. If $H,K \in D$ and $\{e\} = E(H) \bigcap E(K)$, then $e$ is a *marker* edge of $H$ and $K$. If $K = p(H)$ (recall p(·) denotes "parent of"), then $e$ is a *child marker* of $K$ and the *parent marker* of $H$, in this latter capacity denoted $pm(H)$. We assume that $root(D) = root(T)$ is chosen to have at least one edge not in any other node. Picking one such edge arbitrarily, say $m$, we define $pm(root(D)) = m$. In our algorithm we *always chose as m* the edge corresponding to the first, by assumption singleton, column of $M$. This choice guarantees that $m$ is never in "$P$" (see the definition of the hypopath problem) and that $root(D)$ is always a bond given that $D$ is a "$t$-decomposition," and thus simplifies the statement of (R5) in section 5.

Let $e = pm(H)$ have ends $u_1$ and $u_2$ in $H_1$ and $v_1$ and $v_2$ in $K = p(H)$ (so, $H \neq root(D)$). An *orientation* of $H$ with respect to $K$ is a bijection $f : \{u_1, u_2\} \rightarrow \{v_1, v_2\}$. Given $f$, $H$ is said to be *oriented* with respect to $K$. To *reorient* $H$ is to change $f$. The decomposition $D$ is *oriented* if every child member is oriented with respect to its parent. The collection of these orientations is referred to as the *orientation* of $D$.

An oriented decomposition $D$ is a *t-decomposition* if (i) every member of $D$ has at least three edges and is a polygon, bond or prime; (ii) only bond members of $D$ have edges parallel to their parent marker; and (iii) no two polygons or bonds have a marker edge in common.

To *relink* a polygon member of $D$ is to reorder its edges, thus producing a 2-isomorphic copy.

Corresponding to the above decomposition there is a composition. Let $D$ be an oriented decomposition with $H,K \in D$, $K = p(H)$, and $e = pm(H)$. Let $e$ have ends $u_1$ and $u_2$ in $H$. To *merge* $H$ with $K$ is to delete $e$ from both graphs and identify $u_i$ with $f(u_i)$ $(i = 1, 2)$, where $f$ is the orientation of $H$ with respect to $K$. The resulting graph is denoted $m_f(H,K)$, or simply

$m(H,K)$. Note that $m(H,K)$ is uniquely determined up to the names of the nodes being identified.

Replacing, in $D$, the graphs $H$ and $K$ with $m(H,K)$ yields a new decomposition $D'$ having one fewer member, where the orientation of $D$ induces an orientation of $D'$. Repeating this process an additional $|D|-2$ times yields a single graph denoted $m(D)$. It is straightforward to verify that $m(D)$ does not depend on the order in which the members of $D$ are merged. The graph $m(D)$ is the *merged* graph of $D$.

Let $D = \{Q, H_1, \ldots, H_t\}$ be an oriented decomposition with $Q = p(H_i)$ $(1 \leq i \leq t)$. Let $f_1, \ldots, f_t$ be the orientations of $H_1, \ldots, H_t$ respectively. Then $Q_f\,[H_1, \ldots, H_t]$ denotes $m(D)$, where $f$ refers to $f_1, \ldots, f_t$.

(4.1) **Theorem.** Let $G$ and $G'$ be 2-isomorphic nonseparable graphs and let $G = Q_f\,[H_1, \ldots, H_t]$. Assume that $Q$ has at least 3 edges, is a polygon, bond, or prime, and that if $Q$ is a polygon, then for each $H_i$, $H_i \backslash \{pm(H_i)\}$ is nonseparable. Then there exist graphs $Q', H_1', \ldots, H_t'$ and an orientation $f'$ such that $Q$ is 2-isomorphic to $Q'$, $H_i$ is 2-isomorphic to $H_i'$ $(1 \leq i \leq t)$ and $G' = Q'_f\,[H_1', \ldots, H_t']$. Further, if $Q$ is not a polygon, then $Q' = Q$.

**Proof.** This may be deduced from Theorem 2.1 and results in [6]. In particular, we may replace $Q$ and each of the $H_i$ $(1 \leq i \leq t)$ by its "standard" decomposition in the sense of [6]. Now, the condition that when $Q$ is a polygon, each $H_i \backslash \{pm(H_i)\}$ is nonseparable, implies that these decompositions taken together form a "standard" decomposition $D$ of $G$ except for some possible adjacent bonds. But it is easy to see that merging these bonds does not affect the class of graphs obtainable as the merged graph by first reorienting $D$ and then merging. On the other hand, because $G$ and $G'$ have the same cycles, it follows from the uniqueness result Theorem 18 of [6] that merging the adjacent bonds in $D$ does yield a "standard" decomposition of $G'$. The theorem now follows by applying Theorem 2.1 and using the easy fact that any 2-isomorphism of a graph can be carried out as a 2-isomorphism on the simplification followed by the reinclusion of the deleted edges.

●

Let $P \subseteq E(m(D))$. We consider the following ***hypopath problem:*** Given a t-decomposition $D$ and a set $P \subseteq E(m(D))$, find a $t$-decomposition $D'$, if one exists, such that $m(D')$ is 2-isomorphic to $m(D)$ and $P$ is a path of $m(D')$.

Define $\hat{D}$, the ***reduced t-decomposition*** of $D$ with respect to $P$, to be the minimal decomposition contained in $D$ and containing all members of $D$ that have an edge of $P$. Clearly, $P$ is a hypopath of $m(D)$ if and only if $P$ is a hypopath of $m(\hat{D})$. Note that the arborescence structure of the original $t$-decomposition $D$ induces an arborescence structure on $\hat{D}$, but it need not be the case that $root(\hat{D}) = root(D)$.

The following classification scheme is crucial to our method. Let $H$ be a graph, $m$ a distinguished edge ($m$ is typically a parent marker) and $\phi \neq X \subseteq E(H) - \{m\}$. Consider the following mutually exclusive ***arrangements*** for $(H,X,m)$:

(1) $X \bigcup \{m\}$ is a cycle;

(2) $X$ is a path with $m$ incident to one end-node and one internal node;

(3) $X \bigcup \{m\}$ is a path with $m$ an end-edge;

(4) $X \bigcup \{m\}$ is a path with $m$ not an end-edge.

If $(H,X,m)$ satisfies (i), set $A(H,X,m) = i$; otherwise set $A(H,X,m) = 5$. Define the ***type*** of $(H,X,m)$ by $T(H,X,m) = \min\{A(H',X,m): H' \text{ is 2-isomorphic to } H\}$. $(H,X,m)$ is ***good*** if $A(H,X,m) = T(H,X,m) < 5$.

(4.2) **Lemma.** If $T(H,X,m) = i$, then $A(H,X,m) \geq i$. If $T(H,X,m)=1$, then $A(H,X,m)=1$. If $T(H,X,m)=2$, then $A(H,X,m) = 2$ or $5$.

**Proof.** The result follows easily from the definition and the easy part of Theorem 2.1.

●

For the remainder of this section the following notation is used. $\hat{D}$ is the reduced $t$-decomposition of $D$ with respect to $P$, and $Q \in \hat{D}$ is fixed. Each child of $Q$ is the root of a unique maximal $t$-decomposition contained in $\hat{D}$. Let $S_1,\ldots,S_t$ be these $t$-decompositions and let $H_i = m(S_i)$ $(1 \leq i \leq t)$. $H_1,\ldots,H_t$ are the ***complete children*** of $Q$. Let $m = pm(Q)$ and define

$H = Q_f[H_1,\ldots,H_t]$ where $f$ is the orientation induced by $D$. Define $m_i = pm(H_i)$ $(1 \le i \le t)$, $X = P \bigcap E(H)$ and $W_Q = (P \bigcap E(Q)) \bigcup \{m_1,\ldots,m_t\}$. Note that if $Q$ is a polygon, then $H_i \backslash \{m_i\}$ $(1 \le i \le t)$ is nonseparable because $root(S_i)$ cannot be a polygon. Hence, Theorem 4.1 applies.

Suppose $F$ is a $t$-decomposition contained in $D$ with $K = m(F)$ and $P \bigcap E(K) \neq \phi$. In what follows $T(K) := T(K, P \bigcap E(K), pm(root(F)))$, $A(K) := A(K, P \bigcap E(K), pm(root(F)))$ and $K$ *good* means $T(K) = A(K) < 5$.

The following three lemmas provide some basic results. The proofs are straightforward and are left to the reader. An *end-node* of $H[X]$ is a node incident to exactly one edge of $X$.

(4.3) **Lemma.** Suppose $X$ is a disjoint union of paths of $H$.

(a) If $T(H_i) > 1$ (respectively, $> 3$) (some $i$), then $H_i$ contains an end-node (respectively, two end-nodes) of $H[X]$ not incident to $m$, and $T(H) > 1$ (respectively, $> 3$).

(b) If $H[X]$ has an end-node (respectively, two end-nodes) in $H_i$ (some $i$) and not incident to $m_i$, then $T(H_i) > 1$ (respectively, $> 3$).

●

By Lemma 4.3, $T(H) \le 4$ implies at most two of the $(H_i, X_i, m_i)$ are not type 1. Further, if for some $i$, $T(H_i) = 4$, then $T(H_j) = 1$ for $j \neq i$.

(4.4) **Lemma.** If $A(H) \le 4$, then $A(H_i) \le 4$ $(1 \le i \le t)$. Moreover, if $T(H_i) = 5$ for some $i$, then $P$ is not a hypopath of $m(\hat{D})$.

●

(4.5) **Lemma.** Let $H_i'$ be a 2-isomorphic copy of $H_i$ for some $i$. Assume $A(H_i') = A(H_i)$. Then there is an orientation of $H_i'$ such that replacing $H_i$ by $H_i'$ leaves $A(H)$ unchanged.

●

Theorems 4.6 and 4.8 are the heart of the algorithm for solving the hypopath problem.

(4.6) **Theorem.** Suppose that $Q$ is not a polygon. If each $H_i$ is good, then either there exists an orientation $f'$ such that $Q_{f'}[H_1,\ldots,H_t]$ is good, or $T(H) = 5$.

**Proof.** Assume $T(H) \leq 4$. Then by Theorem 4.1, since $Q$ is not a polygon, there are 2-isomorphic copies $H_1', \ldots, H_t'$ of $H_1, \ldots, H_t$, respectively, and an $f'$ such that $A(H') = T(H) \leq 4$, where $H' = Q_{f'}[H_1', \ldots, H_t']$. By Lemma 4.4, $A(H_i') \leq 4$ $(1 \leq i \leq t)$, and by Lemma 4.5 we may assume not all $H_i'$ are good. Suppose $H_1'$ is not good. Then by Lemma 4.2, $T(H_1) = 3$ and $A(H_1') = 4$, and so by Lemma 4.3, $T(H_i) = 1$ $(i > 1)$. Now, it is obvious that $A(H') \geq 4$, and that replacing $H_1'$ by $H_1$, in either orientation, does not decrease $A(H')$.

●

(4.7) *Computational Remark.* $f'$ is straightforward to compute. This remark applies as well to Theorems 4.8-4.10. In particular, by Lemma 4.3, either exactly one $H_i$ is of type 4, and all others are of type 1, or at most two have type 2 or 3 and all others are type 1. In the first case no reorientation is needed. In the second case there are only two possible orientations for the non-type 1 $H_i$, and in determining which of these are good only the degrees of the ends of $pm(H_i)$ in $H_i[P \bigcap E(H_i)]$ are needed. $f'$ can thus be found in time linear in $|W_Q|$.

●

A similar theorem holds when $Q$ is a polygon, but because polygons may have nontrivial 2-isomorphisms a slight modification is needed. Let $b$ be the number of non-type 1 complete children of $H$. By Lemma 4.3 it may be assumed that $b \leq 2$. Let $Z \subseteq W_Q$ be the set of child markers of $H$ that correspond to the non-type 1 complete children. Assume that either $Z = \phi$, $Z = \{m_1\}$ or $Z = \{m_1, m_2\}$. The appropriate relinking procedure is:

**Procedure** RELINK1($Q$)

$Q$ must be a polygon, and $Z$ must satisfy $|Z| \leq 2$. Relink $Q$ so that $W_Q - Z$ is a path, one end of $W_Q - Z$ is incident to $m$, the other end is incident to $m_1$ (if it exists), and $m_2$ (if it exists) is incident to $m$.

●

(4.8) **Theorem.** Suppose RELINK1($Q$) has been applied. If each $H_i$ is good, then either there exists an orientation $f'$ such that $Q_{f'}[H_1, \ldots, H_t]$ is good or $T(H) = 5$.

**Proof.** There are essentially four cases. If $Z = \phi$, the result is clear.

If $Z=\{m_1\}$ and $|W_Q| < |E(Q)|-1$, then $T(H)\geq 3$. If $A(H_1) = 2$ or 3, then it is easily seen that there exists an $f'$ such that $A(Q_{f'}[H_1,...,H_t])=3$. If $A(H_1)=4$, then Theorem 4.1 implies $T(H)=5$ since $T(H)\geq 4$ by Lemma 4.3 and $W_Q$ does not have enough edges for $H[X\bigcup\{m\}]$ to be connected.

If $Z=\{m_1\}$ and $|W_Q|=|E(Q)|-1$, then $T(H)\geq 2$. If $A(H_1) = 2$, 3 or 4, then clearly $f'$ exists such that $A(Q_{f'}[H_1,...,H_t])=A(H_1)$, and so if $A(H_1)=2$ we are done. In the case $A(H_1) = 3$, the easy part of Theorem 2.1 implies $T(H)\geq A(H_1)$ since $X\bigcup\{m\}$ does not contain a cycle, and when $A(H_1)=4$, Lemma 4.3 implies $T(H)\geq A(H_1)$.

If $Z=\{m_1,m_2\}$ and $|W_Q| < |E(Q)|-1$, then $T(H)\geq 4$. On the other hand, by Lemma 4.3, $T(H)\leq 4$ only if $T(H_i)\leq 3$ $(i=1,2)$. In this case $T(H)=4$ is clearly achievable.

Suppose $Z=\{m_1,m_2\}$ and $|W_Q|=|E(Q)|-1$. Then $T(H)\geq 4$. If $T(H_1)=T(H_2)=2$, then $T(H)=5$ since $X\bigcup\{m\}$ contains a cycle. If $T(H_i)\geq 4$ $(i=1,2)$, then $T(H)=5$ by Lemma 4.3. In all other cases it is easy to see $T(H)=4$ is achievable.

●

**Algorithm** TYPING

*Input:* The reduced $t$-decomposition $\hat{D}$ corresponding to a $t$-decomposition $D$ and a nonempty set $P\subseteq E(m(D))$; the ***depth partition*** $\pi=(\pi_0,...,\pi_s)$ of $\hat{D}$, where $G\in\pi_j\,(0\leq j\leq s)$ if the unique path from $G$ to $root(\hat{D})$ has $j$ arcs. We assume $s\geq 1$.

*Output:* The conclusion that $P$ is not a hypopath of $m(D)$, or a reoriented $\hat{D}$, and hence a reoriented $D$, such that each complete child $H_i$ of $root(\hat{D})$ is good and $T(H_i)$ is known.

*Comment.* Steps T1 and T3 are modified in section 5 under "Rules for selecting $K_1$,$K_2$."

*Step T1.* For each polygon $H\in\pi_s$ apply RELINK1($H$). If $T(H)=5$ for some $H\in\pi_s$, stop – Lemma 4.4 implies $P$ is not a hypopath of $m(D)$. (Note that each $H\in\pi_s$ is a polygon, bond, or prime, and so the computation of $T(H)$ is easy. The data structures required to do this computation in time linear in $P\bigcap E(H)$ are discussed in section 6.) Set $i\leftarrow s-1$.

*Step T2.* If $i=0$, stop -- the desired $\hat{D}$ has been found.

*Step T3.* Let $Q\in\pi_i$, let $H_1,...,H_t$ be the complete children of $Q$ in $\hat{D}$, and let $H=Q_f[H_1,...,H_t]$ where $f$ is the current orientation of $\hat{D}$. If more than two of the $H_i$ are not type 1, stop -- Lemma 4.3 implies $P$ is not a hypopath of $m(\hat{D})$. If $Q$ is a polygon, apply RELINK1($Q$). Find, if possible, an orientation $f'$ and a corresponding new $\hat{D}$ such that $Q_{f'}[H_1,...,H_t]$ is good -- see (4.7). If $T(H)=5$, stop.

Repeat the above procedure until a stop occurs, or every $Q\in\pi_i$ has been treated. When the latter occurs, set $i\leftarrow i-1$ and go to Step T2.

●

The considerations required are slightly different when $Q = root(\hat{D})$ (although the proofs are essentially the same). Typing is no longer important; we wish rather to determine whether or not $P$ can be made into a path. For example, in this case $T(H)=5$ is preferred to $T(H)=4$.

(4.9) **Theorem**. Suppose $Q = root(\hat{D})$ and $Q$ is not a polygon. If each $H_i$ is good, either there exists an orientation mapping $f'$ such that $P$ is a path of $Q_{f'}[H_1,...,H_t]$, or $P$ is not a hypopath of $H$.

●

For polygons the following relinking procedure is needed.

**Procedure** RELINK2($Q$)

$Q$ must be a polygon. Relink $Q$ so that $W_Q - Z$ is a path and, if $m_i$ exists ($i=1,2$), so that $m_i$ is incident to an end of the path $W_Q - Z$.

●

(4.10) **Theorem**. Suppose $Q = root(\hat{D})$ and that RELINK2($Q$) has been applied. If each $H_i$ is good, then either there exists an orientation $f'$ such that $P$ is a path of $Q_{f'}[H_1,...,H_t]$, or $P$ is not a hypopath of $H$.

●

**Algorithm** HYPOPATH

*Input:* A $t$-decomposition $D$ and a set $P \subseteq E(m(D))$.

*Output:* A (new) $t$-decomposition $D$ such that $P$ is a path of $m(D)$ and $m(D)$ is 2-isomorphic to the input $m(D)$, or the conclusion that no such 2-isomorphism exists, i.e., that $P$ is not a hypopath of $D$.

*Step H1.* Compute the reduced $t$-decomposition $\hat{D}$ for $P$. If $\hat{D}$ has just one member, let this member be $Q$ and go to Step H3.

*Step H2.* Apply TYPING. If this results in the conclusion that $P$ is not a hypopath, stop; otherwise, set $Q \leftarrow root(\hat{D})$.

*Comment.* Step H3 is expanded in section 5 under "Rules for selecting $K_1,K_2$."

*Step H3.* Let $H_1, \dots, H_t$ be the complete children of $Q$ in $\hat{D}$. If more than two $H_i$ are not of type 1, stop -- $P$ is not a hypopath. Apply RELINK2($Q$) if $Q$ is a polygon. Find, if possible, an orientation $f'$ and corresponding $\hat{D}$ such that $P$ is a path of $Q_{f'}[H_1,...,H_t]$ -- see (4.7).

●

## 5. DECOMPOSITION UPDATE

Assume the following are given: A $t$-decomposition $D$, a path $P$ of $m(D)$, and the reduced $t$-decomposition $\hat{D}$ of $D$ with respect to $P$. Let $C$ be a set such that $C \bigcap E(m(D)) = P$ with $C - P \neq \phi$. The purpose of this section is to give a method to determine a $t$-decomposition $D^*$ for the graph obtained from $m(D)$ by adding the edges of $C-P$ so that $C$ is a cycle, and $C-P$ is incident to $m(D)$ at exactly two nodes. (In terms of the notation in section 3, $P = P_{k+1}$, $D = D_k'$, $C = C_{k+1}$ and $D^* = D_{k+1}$.)

The idea for finding $D^*$ is intuitively simple. First we select appropriate graphs $K_1, K_2 \in D$ containing the ends of $P$. If it happens that $K_1 = K_2$, then the construction of $D^*$ changes $D$ very little. When $K_1 \neq K_2$, more complicated modifications are involved. Where $R$ is the unique path in $D$ between $K_1$ and $K_2$, we first "squeeze" off certain parts of any polygons that occur in $R$. Then the remaining graphs are merged and $C-P$ appropriately added.

We introduce a convention. For a given node $x$ of $m(D)$, there may be several nodes of members of $D$ that are merged into $x$. In the remainder of this section we identify all these nodes with the name $x$.

Using the following rules, the graphs $K_1, K_2$ are conveniently found while executing TYPING and HYPOPATH.

### RULES FOR SELECTING $K_1, K_2$

*Comment.* The end-nodes of $P$ contained in $K_1, K_2$ are computed while applying rules (R1)-(R4). They will be encountered naturally during the calculations in Steps T1, T3 and H3.

*Additions to Steps T1 & T3 in TYPING:*

(R1) If $T(H_i) = 1$ (all $i$) and $T(H) = 2$ or 3, either set $K_1 \leftarrow Q$ if $K_1$ is not yet assigned, or set $K_2 \leftarrow Q$.

(R2) If $T(H_i) = 1$ (all $i$) and $T(H) = 4$, set $K_1, K_2 \leftarrow Q$.

(R3) Suppose $T(H_k) = 2$ or 3 for some $k$, $T(H_i) = 1$ (all $i \neq k$) and $T(H) = 4$. Let $K_2$ be the node on the unique path in $D$ between $Q$ and $K_1$ that is nearest $K_1$ and contains the same end of $P$ as $Q$.

*Additions to Step H3 of HYPOPATH:*

(R4) If $K_1$ has not been assigned, set $K_1, K_2 \leftarrow Q$. Otherwise, let $K_2$ be the node on the unique path in $D$ between $Q$ and $K_1$ that is nearest $K_1$ and contains the same end of $P$ as $Q$.

(R5) Suppose $K_1, K_2$ have been selected and $K_1 = K_2$. If the ends of $P$ are the ends of $pm(K_1)$ and $K_1$ is not a bond, set $K_1, K_2 \leftarrow p(K_1)$. If the ends of $P$ are the ends of another marker $m_i$ of $K_1$, $K_1$ is a polygon, and $m_i$ is an edge of a bond $G$, set $K_1, K_2 \leftarrow G$.

●

The application of the final rule (R5) is needed for the proof of Lemma 5.4.

In the above list of rules, (R1) is justified by Lemma 4.3. In particular, (4.3a) implies $N(H)$ has an end-node $x$ of $P$ not incident to $pm(Q)$, and (4.3b) implies $x \in N(Q)$. (R2) is justified similarly. For (R3), first note that $K_1$ has already been selected because (R1) must have been applied to $H_k$ or one of its descendants. Now by Lemma 4.3, $P$ has an end $x \in N(Q)$ not incident to $pm(Q)$, which implies that $K_2$ could not previously have been assigned. However, $x$ may be incident to a child marker of $Q$, in which case $K_2 \neq Q$ is possible. Consider now $(R4)$. Clearly $N(H)$ contains both ends of $P$. If neither $K_1$ nor $K_2$ has been assigned, then $T(H_i) = 1$ (all $i$). Hence, Lemma 4.3 implies both ends of $P$ are in $N(Q)$, as required. Similar reasoning justifies the choice of $K_2$ when $K_1$ has already been assigned.

(5.1) **Lemma.** The application of (R1) during the execution of HYPOPATH and TYPING finds $K_1, K_2 \in D$ such that $u_i \in N(K_i)$ $(i = 1,2)$ where $u_1, u_2$ are the distinct end-nodes of $P$ in $m(D)$. Where $R$ is the unique path in $D$ between $K_1$ and $K_2$, no internal node of $R$ contains $u_1$ or $u_2$.

**Proof.** By the remarks preceeding the lemma we need only prove the last statement. There is nothing to prove if $K_1 = K_2$. If (R3) or (R4) was applied we use the fact that $u_1$ is not incident to $pm(K_1)$, and so is contained only in descendants of $K_1$. The only remaining possibility is that (R1) has been applied twice. But by (4.3a), in this case neither $K_j$ is a descendant of the other, and $u_j$ $(j = 1, 2)$ is contained only in descendants of $K_j$. This completes the proof.

●

Given that $K_1, K_2$ have been selected, we can carry out the actual update. Recall that $t$-decompositions have an underlying arborescence structure, and are oriented. In what follows, when new members are added to $D$ to form $D^*$ we will not explicitly specifiy the new root, how new arcs of the arborescence are directed, or how the orientations of new members are determined. The specification of the new root is straightforward: It will always be the unique node that contains the parent marker of the old root. Given a root, the directions of the arcs of $D^*$ are uniquely specified. Now, for the orientation of $D^*$, the case $K_1 = K_2$ is easy, and when $K_1 \neq K_2$, with one exception, we employ the natural orientation; thus, in procedure SQUEEZE we choose the orientation such that the merged graph is the one we started with. The one exception occurs in Step U0 where the orientation of "$\{f\} \bigcup (C-P)$" is arbitrary, and in any case can only be determined once "$f$" has been added to another member of $D^*$.

We introduce the following procedure for application in Step U2 of UPDATE. Note that $D^*$ and $R$ are defined in UPDATE.

**Procedure** SQUEEZE($L$)

> $L$ must be a path in some polygon $S$ of $R$. If $L$ has fewer than two edges, do nothing. Otherwise, let $f'$ be a new element and replace $S$ in $D^*$ by the two polygons formed by adding $f'$ to the paths $L$ and $S-L$, respectively. Replace $S$ in $R$ by the polygon formed from $S-L$.
>
> ●

**Algorithm** UPDATE

*Input:* A $t$-decomposition $D$, a path $P$ of $m(D)$, and a set $C$ such that $C \bigcap E(m(D)) = P$ and $C-P \neq \phi$; nodes $K_1, K_2$ of $D$ and nodes $u_1, u_2$ of $K_1, K_2$, respectively, such that the conclusions of Lemma 5.1 hold. We also assume that (R5) has been applied.

*Output:* A $t$-decomposition $D^*$ of the graph obtained from $m(D)$ by adding the edges of $C-P$ so that $C$ is a cycle, and $C-P$ is incident to $m(D)$ at exactly two nodes.

> *Step U0.* Set $D^* \leftarrow D$. If $|C-P|=1$ set $\{f\} \leftarrow C-P$; otherwise, let $f$ be a new element, form a polygon with edge-set $\{f\} \bigcup (C-P)$ (the order of the edges is irrelevant) and add this polygon to $D^*$. If $K_1 = K_2$, go to Step U1; otherwise, go to Step U2.
>
> *Step U1.* (Case $K_1 = K_2$) Apply the appropriate one of (U1.1)-(U1.3) and stop:
>
> (U1.1) If $K_1$ is not a polygon, join $u_1$ and $u_2$ by $f$ in $K_1$.
>
> (U1.2) Suppose $K_1$ is a polygon and $u_1, u_2$ are joined in $K_1$ by edge $f'$. Let $f''$ be a new element. Replace $f'$ by $f''$ in $K_1$ and add a bond with edge-set $\{f, f', f''\}$ to $D^*$.
>
> (U1.3) Suppose $K_1$ is a polygon and $u_1, u_2$ are not adjacent. Then there are two distinct paths in $K_1$, say $L_1$ and $L_2$, joining $u_1$ and $u_2$. Let $f_1, f_2$ be new elements. Delete $K_1$ from $D^*$, add polygons formed by joining the ends of $L_i$ with $f_i$ $(i=1,2)$, and add a bond with edge-set $\{f, f_1, f_2\}$.

*Step U2.* (Case $K_1 \neq K_2$) Let $R$ be the unique path $K_1 = J_1, \ldots, J_{s+1} = K_2$ in $D^*$ joining $K_1$ and $K_2$. Let $\{m_j\} = E(J_j) \bigcap E(J_{j+1})$ $(1 \leq j \leq s)$. Apply (U2.1)-(U2.4) and stop:

(U2.1) Suppose $1 < j \leq s$, $J_j$ is prime and $\{m_{j-1}, m_j\}$ is a cycle of $J_j$, or $J_j$ is a bond on at least 4 edges and $p(J_j)$ is not in $R$. Let $f'$ be a new element. Let $J_j'$ be $J_j$ with $\{m_{j-1}, m_j\}$ deleted and $f'$ added (with the same ends), and let $B$ be a bond with edge-set $\{m_{j-1}, f', m_j\}$. Replace $J_j$ in $R$ by $B$; delete $J_j$ from $D^*$ and replace it by $J_j'$ and $B$. (Note: There can be at most one $J_j$ as above.)

(U2.2) If $K_1$ is a polygon, let $L_1, L_2$ be the two paths joining $m_1$ and $u_1$. Apply SQUEEZE($L_i$) $(i = 1, 2)$. Do the same for $K_2$, if $K_2$ is a polygon.

(U2.3) For each internal $J_j$ in $R$ that is a polygon, let $L_1, L_2$ be the two components of $J_j - \{m_j, m_{j+1}\}$ and apply SQUEEZE($L_i$) $(i = 1, 2)$.

(U2.4) Let $G$ be $m(R)$ with $f$ joining $u_1$ and $u_2$. Delete $R$ from $D^*$ and add $G$.

●

We must now prove that $D^*$ is a $t$-decomposition and that it is the "right" $t$-decomposition. The second is the easier of the two assertions, and is treated in the following lemma.

(5.2) **Lemma.** After UPDATE has been applied, $m(D^*)$ equals the graph obtained from $m(D)$ by adding the edges $C - P$ so that $C$ is a cycle and $C - P$ is incident to $m(D)$ at exactly two nodes.

**Proof.** This proof reduces to a careful reading of the steps of UPDATE. We treat only the case $|C - P| > 1$. In Step U0 denote $\{f\} \bigcup (C - P)$ by $S'$. Note that $S'$ is a member of $D^*$. Let $D_1^*$ denote the final $D^*$ with $S'$ deleted and $f$ deleted from the remaining member of $D^*$ that contains it as an edge.

First, we observe that $m(D_1^*) = m(D)$. Step U0 and (U1.1) present no difficulties. In (U1.2), once $f$ is deleted, merging the remaining 2-bond $\{f', f''\}$ with $K_1$ yields $D$; in (U1.3), merging the remaining 2-bond $\{f_1, f_2\}$ with the $L_1$ and $L_2$ polygons yields $D$. Now consider Step U2. Deleting $f$ in this case just means leaving out a part of (U2.4). But then it is straightforward to see that none of (U2.1)-(U2.4) change $m(D^*)$. This proves $m(D_1^*) = m(D)$.

Now to complete the proof, note that $f$ is added in either (U1.1)-(U1.3) or (U2.4). It is clear in (U1.1) and (U2.4) that $f$ joins $u_1$ and $u_2$ in $m(D) = m(D_1^*)$. Steps (U1.2) and (U1.3) are handled as in the previous paragraph. Now, given that $f$ has been added, it only remains to reinsert $S'$. But clearly this has the desired effect of adding $C - P$ as a path meeting $m(D)$ at $u_1, u_2$.

●

The main step in proving that $D^*$ is a $t$-decomposition is the following lemma.

(5.3) **Lemma.** $G$ in (U2.4) is prime.

**Proof.** Let $x$ and $y$ be distinct nodes of $G$. We show that there are three node disjoint (except for $x$ and $y$) paths joining $x$ and $y$. Assume that in (U2.4), $x \in N(J_i)$ and $y \in N(J_k)$ where $i \leq k$. We also assume $\{x,y\} \bigcap \{u_1,u_2\} = \phi$; the proof when this latter assumption fails is an easy variation of the following arguments.

By (U2.2) and (U2.3) we may assume neither $J_i$ nor $J_k$ is a polygon. If $i=k$, then since $J_k$ is either prime or a bond with at least three edges, it contains three node-disjoint paths from $x$ to $y$. Suppose one of these paths $L$ contains $m_i$. Then the merge of $J_{i+1}, \dots, J_s$, since this graph is 2-connected, contains a cycle $C$ containing $m_i$, and $(C \bigcup L) - \{m_i\}$ is a path in $G$. Applying this procedure at most twice, we obtain the desired paths in $G$.

Suppose that $i \neq k$. Consider $J_i$. If $i \neq 1$ and $J_i$ is prime, $J_i[\{m_{i-1}, m_i\}]$ has at least three nodes by (U2.1) and there are three node-disjoint paths from $x$ to three of these nodes, including the ends of $m_i$, and such that $m_i$ and $m_{i-1}$ are not in these paths. Let $\alpha_1$ be the end of $m_{i-1}$ used and let $\alpha_2$ be the other end of $m_{i-1}$. If $i=1$, find three node-disjoint paths in $J_1$ from $x$ to $u_1$ and the ends of $m_1$ (and not containing $m_1$); in this case $\alpha_2$ is undefined and $\alpha_1 = u_1$. In the case that $J_i$ is a bond, let $\alpha_1 = x$ and $\alpha_2$ be the other node in $J_i$. Perform the symmetric construction for $J_k$ defining $\beta_1$ and $\beta_2$. Let $L_1$ be a path in the merge of $J_1, \dots, J_{i-1}$ from $u_1$ to $\alpha_1$, avoiding $\alpha_2$. $L_2$ is defined similarly for $u_2, \beta_2$. Finally, let $C$ be a cycle of the merge of $J_{i+1}, \dots, J_{k-1}$ (empty if $i+1=k$) containing $m_i$ and $m_{k-1}$. Then $L_1 \bigcup L_2 \bigcup \{f\} \bigcup (C - \{m_i, m_{k-1}\})$ together with the paths constructed in $J_i$ and $J_k$ gives the desired three paths in $G$.

●

We are now prepared to prove

(5.4) **Lemma.** $D^*$ is a $t$-decomposition.

**Proof.** We must verify that $D^*$ has the following properties: (a) It is a decomposition, (b) every member has at least three edges, (c) there are no adjacent polygons or bonds, (d) no non-bond has an edge parallel to its parent marker, and (e) every member is a bond, polygon, or prime.

Most of the work in verifying (a)-(e) involves simply a careful reading of UPDATE, as in the proof of Lemma 5.2. We mention only the highlights.

For (a) note that in Step U0 and in the case $|C-P| > 1$, the conditions for $D^*$ to be a decomposition are violated since the polygon $(C-P)\bigcup\{f\}$ has no edges in common with any other member of $D^*$. However, it is easy to verify that $f$ is later added to exactly one other member of $D^*$, thus restoring the required property.

In proving (b) note that $|C-P| > 1$ implies $|(C-P)\bigcup\{f\}| \geq 3$ in Step U0. Also note that whenever SQUEEZE($L$) is applied, no change occurs unless $L$ has at least two edges ($S-L$ always has at least two edges). Finally, note that since $J_j$ in (U2.1) is prime, or a bond on at least 4 edges, $|E(J_j)|-1 \geq 3$.

Part (e) is easy to verify in view of Lemmna 5.3. Consider now (c). Step U0 creates no new adjacencies, and so no violations to (c). However, a polygon may be created and it must then be verified that edge $f$ is added to no other polygon. That this is true follows from inspection and Lemma 5.3. Note also that no violation of (c) is created in (U1.2). Here $f'$ cannot be in another bond of $D^*$ by (R5).

Finally, consider (d). (U1.1) preserves (d) by (R5). Steps (U1.2), (U1.3), and (U2.1)-(U2.3) increase the "edge multiplicities" only in bonds. In (U2.4), $u_1$ and $u_2$ are not adjacent until $f$ is added, and otherwise edge multiplicities are not increased. Hence, if in (U2.4) the parent marker of $G$ is parallel to another edge, then it was the parent marker of a bond in $R$ with at least 4 edges. This possibility is ruled out by (U2.1). This completes the proof.

●

## 6. THE MAIN ALGORITHM

We begin by stating GRAPH. This is followed by a listing of the data structures that are needed to implement the algorithm, and a discussion of how these data structures are updated.

**Algorithm** GRAPH

*Input*: An $r \times c$ totally nonseparable $\{0,1\}$-matrix $M$. We assume (see the first paragraph of section 4) that the first column of $M$ is a singleton. (Such a column can always be added without affecting the realizability of $M$.)

*Output*: The conclusion that $M$ is not realizable, or a realizing graph $G$ for $M$. We use the notation introduced in section 3.

> *Step G1.* $M_1$ is realized by a bond $G_1$ with two edges. Set $D_1 \leftarrow \{G_1\}$ where for $pm(G_1)$ we choose the non-tree edge of $G_1$. (Note that $D_1$ is a decomposition, but not a $t$-decomposition because $G_1$ has only two edges. All further $D_j$, $j \geq 2$, will be $t$-decompositions.) If $c = 1$, stop ($M$ is graphic). Otherwise, set $j \leftarrow 2$.
>
> *Step G2.* Set $P \leftarrow P_j$ and $D \leftarrow D_{j-1}$. Apply HYPOPATH. If $P$ is not a hypopath of $m(D)$, stop.
>
> *Comment.* If a maximal graphic subset of columns is desired, not just a determination of the realizability of $M$, then instead of terminating when $P$ is not a hypopath, simply discard column $j$ and continue the algorithm.
>
> *Step G3.* Set $C \leftarrow C_j$ and apply UPDATE. (Note that the quantities $K_1, K_2, u_1, u_2$ required for input to UPDATE are calculated in HYPOPATH, and the associated calls to TYPING, through application of rules (R1)-(R5).) Set $D_j \leftarrow D^*$.
>
> *Step G4.* If $j = c$, set $G \leftarrow m(D_j)$ and stop; otherwise set $j \leftarrow j+1$ and go to Step G2.

●

The validity of GRAPH follows from the results of sections 4 and 5.

*DATA STRUCTURES.* Each of the columns and rows of matrix $M$ is assumed to have a distinct name, taken from the integers $1, \ldots, r+c$; all marker edges created during the algorithm are also given distinct names, one for each of the two graphs containing them. Each member of each decomposition is named, and each node of each member is named. These node names for the graphs in any fixed decomposition will always be distinct, even though after merging several may be identified.

(D1) $M$: The given matrix is assumed stored in column list form. Thus, for each column we can access in linear time the list of rows with ones in this column. For each row name we keep an indicator to designate if this row is in a previously processed column.

(D2) ***Members of D, where D is the current t-decomposition***: These are stored and accessed using the standard disjoint set union data structure with path compression (see Tarjan [15]). Thus, for given distinct member names $x, y$ we can perform $makeset(x)$ (assuming $x$ is not yet in any set, create a single element set $\{x\}$ with name $x$), $find(x)$ (find the name of the set containing $x$) and $link(x,y)$ (form the union of the two sets containing $x$ and $y$). The sets are stored as arborescences, the root of which is the name of that set. Computing $find(x)$ involves tracing a directed path from $x$ to the root of its tree, and then "compressing" this path so that all nodes on it point directly to the root. The operation $link(x,y)$ is performed by first performing $find(x)$ and $find(y)$, and completed by creating an arc pointing from the root of the shallower of the two trees to that of the deeper.

(D3) *Nodes of members of $D$*: Disjoint set data structure, as in (D2).

(D4) *Edge locations*: Pointers from edges to the names of the members containing them. Thus, to find the member containing an edge, we first access the member $x$ pointed to by the edge, and then perform a (D2) $find(x)$.

(D5) *Predecessor function $p$ of $D$*: Pointers from member names to member names.

(D6) *Member designation*: For each member name, a indicator designating whether it is a bond, polygon, or prime.

(D7) *Polygon sizes*: The number of edges in each polygon.

(D8) *Polygon edge-sets*: Doubly-linked lists.

(D9) *Edge ends*: Pointers from each edge to its ends. Note that for a given edge, once the two ends $u,v$ have been accessed it will still be necessary to perform two (D3) finds in order to find the current names of these ends. For marker edges, one of the two ends is designated $+$ and one $-$ . This serves to orient $D$.

(D10) *Parent markers*: $pm(x)$ for each member name $x$.

(D11) *Child markers*: For each current member $x \neq root(D)$, $pm(x)$ is associated with some child marker of $find(p(x))$. We keep a pointer from $x$ to this child marker. In the case when $x$ is a bond and $find(p(x))$ is a polygon, we also keep a pointer from the child marker to $x$.

*DATA STRUCTURE APPLICATION AND UPDATE.* We discuss here the highlights of how (D1)-(D11) are applied and updated.

(6.1) Executing Step H1: $P$ and $C-P$ are computed in time $O(|C|)$ using (D1), and the indicators for $C-P$ are updated. Then using (D4) and (D2) we construct a list of members in $D$ containing edges of $P$. This requires $O(|P|)$ (D2) finds. Using this list and (D5), the computation of $\hat{D}$ and simultaneously the depth partition $(\pi_0, \ldots, \pi_s)$ required as input by TYPING uses $O(|\hat{D}|)$ additional (D2) finds.

(6.2) Executing Step T1: If $H$ is a bond, as determined by (D6), this is straightforward. Otherwise, let $X = P \bigcap E(H)$. A list of the elements in $X$ can be conveniently created for each $H \in \hat{D}$ as $\hat{D}$ is computed. If $H$ is prime, then using (D9) and (D3) we construct a representation for $H[X]$ that uses space $O(|X|)$, and do this with $O(|X|)$ (D3) finds.

The most complicated case occurs when $H$ is a polygon and RELINK1 is applied. Starting with the marker $m = pm(H)$, we swap edges in pairs in order to build a path $L$ in $H$ one end of which is $m$, and every other of which is in $X$. At a general step we take the non-$m$ end-edge of $L$ and find its neighbor. If this edge is not in $X$, it is swapped in $H$ with the next edge of $X$ not already in $L$. This uses (D3), (D8) and (D9) and updates (D8) and (D9). It requires time $O(|X|)$ (D3) finds.

(6.3) Executing Steps T3 and H3: In addition to the work in (6.2), (D9) is applied to accomplish the reorientation. See (4.7).

(6.4) Executing (R1)-(R5): The application of (R5) requires special consideration since the final assignment statement may require accessing a child of a node. However, this happens only for polygon-parent, bond-child pairs and (D11) may be used.

(6.5) Executing Step U1: Determining if $K_1$ is a polygon takes constant time because of (D6). (U1.1) takes constant time, as does (U1.2). In (U1.3) particular care must be taken with (D4). Thus, when two polygons are formed from one, then only one of these can retain the name of the old polygon member, and hence for every edge in one of the new polygons (D4) must be updated. In this update we always choose the piece $L_j$ that meets the set $W=(P\bigcap E(K_1))\bigcup F$ where $F$ is the set of child markers of $K_1$ such that P meets the corresponding complete child. It is easy to see then that $L_j\subseteq W$, and hence that $O(|P|)$ work will accomplish the update.

We must also update (D5) for all children of the polygon $L_j\bigcup\{f_j\}$ having the same name. This can be done in $O(|\hat{D}|)$ time since $L_j\subseteq W$ implies all children are in $\hat{D}$.

(6.6) Executing Step U2: First $R$ must be calculated, which uses $O(|R|)$ (D2) finds. (These will be constant-time finds given that the reduced $t$-decomposition for $P$ has already been found in HYPOPATH.) (U2.1) requires four (D3) finds to check adjacency of $m_j$ and $m_{j-1}$. For (U2.2) and (U2.3), the nontrivial work involves the applications of SQUEEZE. Again, by the convention introduced in (6.5), the total work for the SQUEEZEs involves $O(|P|)$ constant-time updates in (D4), and $O(|R|)$ constant-time operations on each of (D2), (D3), (D5)-(D11). Finally, in (U2.4) the forming of $m(R)$ requires $O(|R|)$ (D2) and (D3) links.

(6.7) We remark that (D7) is not necessary to implement the algorithm. However, its availability will sometimes speed up the computation of $T(H)$ when $H$ is a polygon.

## 7. TIME AND SPACE BOUNDS

We begin by giving bounds for one application of HYPOPATH and one application of UPDATE. Note that (R1)-(R5) of section 5 are applied in HYPOPATH.

(7.1) **Lemma**. One application of HYPOPATH uses time bounded by the time required for $O(|P|+|\hat{D}|)$ find operations using (D2) and (D3).

**Proof**. Step H1 takes $O(|P|+|\hat{D}|)$ (D2) finds by (6.1). Now consider Steps T1, T3 and H3. For $Q \in \hat{D}$, let $H_1,\ldots,H_t$ be a list of the complete children of $Q$ in $\hat{D}$ (this list may be empty). Define $W_Q = (P \bigcap E(Q)) \bigcup \{m_1,\ldots,m_t\}$ where $m_i = pm(H_i)$ $(1 \le i \le t)$, and set $H = Q_f[H_1,\ldots,H_t]$. By (6.2) and (6.3), $T(H)$ and an orientation $f'$ such that $Q_{f'}[H_1,\ldots,H_t]$ is good can be found with $O(|W_Q|)$ (D3) finds. But

$$\sum_{Q \in \hat{D}} |W_Q| = |P| + |\hat{D}| - 1.$$

This proves the result.

●

(7.2) **Lemma**. One application of UPDATE uses time bounded by $O(|C|+|R|)$ finds using (D2) and (D3) (R is defined in Step U2).

**Proof**. Step U0 can clearly be carried out in time $O(|C|)$. The remainder of the proof follows from (6.5) and (6.6).

●

For the final derivation of the time and space bounds we need the results (7.3)-(7.11).

(7.3) **Lemma**. Let $G$ be a nonseparable graph with $r \ge 3$ nodes. Let $D$ be a $t$-decomposition of $G$ and let $H_1,\ldots,H_t$ be the prime members of $D$. Then

$$\sum_{i=1}^{t} |N(H_i)| \le 2r-4.$$

**Proof**. The result follows easily by induction on $r$.

●

The algorithm GRAPH proceeds by producing a sequence of $t$-decompositions. Let $D_1,\ldots,D_c$ be the sequence produced for a given $r\times c$ $\{0,1\}$-matrix. For each index $j$ $(1\leq j\leq c)$ such that Step U2 is applied, denote by $R_j$ the path $R$ of nodes merged in (U2.4). Every prime node $H$ that occurs in the algorithm originates from the application of (U2.4) to some $R_j$; $H$ may then be further modified by application of (U1.1) or (U2.1), and so several $H$s may have the same associated $j$, but the value of $j$ for a given $H$ is still well defined. Now define, recursively, the sets $I_H$ by $I_H=\{j\}\bigcup\left\{\bigcup_{J\in R_j,\, J\, prime} I_J\right\}$.

(7.4) **Lemma.** For each prime $H$ that occurs as a node of some decomposition in GRAPH,

$$|N(H)| \geq 3+\sum_{i\in I_H}(|R_i|-\lceil |R_i|/2\rceil).$$

($\lceil x\rceil$ is the least integer not less than $x$.)

**Proof.** Let $p_i=|R_i|$ $(i\in I_H)$. The proof is by induction in $|I_H|$. If $I_H=\{i\}$, then

$$|N(H)| = 2(1-p_i)+\sum_{J\in R_i}|N(J)|.$$

If $J$ is a bond, then $|N(J)|=2$, and if $J$ is not a bond then $|N(J)|\geq 3$. Since no two bonds can share a marker edge, and neither end-graph of $R_i$ can be a bond, by Lemma 5.1, the number of bonds in $R_i$ is bounded by $\lceil p_i/2\rceil-1$. Therefore

$$|N(H)| \geq 2(1-p_i)+3p_i-\lceil p_i/2\rceil+1 = 3+p_i-\lceil p_i/2\rceil .$$

Now let $|I_H|>1$. Define $m=\max\{i: i\in I_H\}$ and $I_H'=I_H-\{m\}$. For each prime node $J\in R_m$, induction yields

$$|N(J)| \geq 3+\sum_{i\in I_J}(p_i-\lceil p_i/2\rceil).$$

Let $R_m'=\{J\in R_m: J\, prime\}$ and $R_m''=R_m-R_m'$. Then

$$|N(H)| \geq 2(1-p_m)+\sum_{J\in R_m'}\left\{3+\sum_{i\in I_J}(p_i-\lceil p_i/2\rceil)\right\}+\sum_{J\in R_m''}|N(J)| .$$

Since $\{I_J: J\in R_m'\}$ partitions $I_H'$, it follows that

$$\begin{aligned} |N(H)| &\geq 2(1-p_m) + 3|R_m'| + \sum_{i \in I_H'} (p_i - \lceil p_i/2 \rceil) + 3(p_m - |R_m'|) - (\lceil p_m/2 \rceil - 1) \\ &= 3 + \sum_{i \in I_H} p_i - \sum_{i \in I_H} \lceil p_i/2 \rceil \end{aligned}$$

●

(7.5) **Theorem**. For $r \geq 3$,

$$\sum_{\text{all } R_i} |R_i| \leq 6r - 12.$$

**Proof**. Let $P$ be the set of prime members of $D_c$. By Lemmas 7.3 and 7.4,

$$\begin{aligned} 6r-12 &\geq 3 \sum_{H \in P} \left\{ 3 + \sum_{i \in I_H} (|R_i| - \lceil |R_i|/2 \rceil) \right\} \\ &\geq \sum_{H \in P} \left\{ 9 + \sum_{i \in I_H} |R_i| \right\} \geq \sum_{\text{all } R_i} |R_i|. \end{aligned}$$

●

Theorem 7.5 says that the total number of graphs ever involved in a (U2.4) merge is $O(r)$. For each $t$-decomposition $D_i$, let $\hat{D}_i$ be the corresponding reduced $t$-decomposition. Then $R_i \subseteq \hat{D}_i$.

Next we bound the number of nodes in $\hat{D}_i$, but not in $R_i$.

(7.6) **Lemma**. Let $H$ be a nonseparable graph and let $X$ be a cycle of $H$. Let $D$ be a $t$-decomposition of $H$ and let $\hat{D}$ be the reduced $t$-decomposition with respect to $X$. Then $|\hat{D}| \leq |X| + 1$.

**Proof**. The theorem is clear if $|\hat{D}| = 1$. If $\hat{D}$ has depth one, then all but at most one node of $\hat{D}$ contains edges of $X$. Again the result is clear. Hence, we may assume that there is a pendant node $K$ of $\hat{D}$ where $K$ is at least distance two from $root(\hat{D})$.

Let $K'$ be the parent of $K$, and let $X' = (X - E(K)) \bigcup \{pm(K)\}$. Clearly $X'$ is a cycle of $m(\hat{D}')$, where $\hat{D}'$ is $\hat{D}$ with $K$ deleted. If $K'$ has more than one child, $\hat{D}'$ is a reduced $t$-decomposition for $X'$. In this case, the theorem follows by induction on $|\hat{D}|$. Suppose $K'$ has only $K$ as a child and that $X \bigcap E(K') = \phi$ (otherwise, again we may apply induction). But then $pm(K)$ is parallel to $pm(K')$ in $K'$ and so $K'$ is a bond.

Now by the choice of $K$, $K'$ is not $root(\hat{D})$; moreover, $p(K')$ is not a bond. Hence $\hat{D}''$,$\hat{D}$ with $K$ and $K'$ deleted, is a reduced $t$-decomposition for $X'$. In addition, note that $|X \bigcap E(K)| \geq 2$; otherwise $K$ has and edge parallel to its parent marker, and is not a bond. Since at least two edges of $X$ have been deleted, and exactly two nodes of $\hat{D}$, the result follows by induction.

●

(7.7) **Theorem**. Let $\hat{D}_1, \ldots, \hat{D}_c$ be the sequence of reduced $t$-decompositions produced by GRAPH. Then

$$\sum_{i=1}^{c} |\hat{D}_i| \leq 2n + c + 6r - 12 .$$

**Proof**. Consider a particular $\hat{D}_i$. Let $R_i = \{K_1\}$ if $K_1 = K_2$ in UPDATE. Now $\hat{D}_i$ can be viewed as a collection of trees attached to $R_i$. For a particular one of these trees $T'$, let $m'$ be the marker linking $T'$ to $R_i$. Then clearly $X = (P_{i+1} \bigcap E(m(T'))) \bigcup \{m'\}$ is a cycle in $m(T')$, and, indeed, $T'$ is a reduced $t$-decomposition for $X$. Hence, by Lemma 7.6, $|T'| \leq |X| + 1 \leq 2\,|X|$. Summing over all such $T'$, we have that the total number of nodes of $\hat{D}_i$ not in $R_i$ is bounded by $2\,|P_{i+1}|$. Hence,

$$\sum_{i=1}^{c} |\hat{D}_i| \leq \sum_{i=1}^{c} (|R_i| + 2|P_{i+1}|) \leq c + 6r - 12 + 2n$$

by Theorem 7.5.

●

(7.8) **Lemma**. $\sum_{|C_j - P_j| \geq 2} |C_j - P_j| \leq 2r .$

●

(7.9) **Lemma**. For a single application of GRAPH, the total number of node names created for nodes of members of $t$-decompositions and the total number of member names created are both bounded by $O(r)$.

**Proof**. For a $t$-decomposition $D$, define $S(D) = \sum(|C| - 3 : C \textit{ is the edge-set of a polygon of } D)$.

Note that in order for (U1.3) or SQUEEZE to be applied it must be that $S(D) > 0$. Moreover, the application of either of these operations reduces $S(D)$. On the other hand, $S(D)$ can only be increased in Step U0, and then only by $|C-P|-2$, whenever $|C-P| \geq 2$. Hence, it follows from Lemma 7.8 that (U1.3) and the SQUEEZE procedure can be applied at most $O(r)$ times.

Now define $S'(D) = \sum(|C| - k(C)$: *$C$ is the edge-set of a polygon of $D$*) where $k(C)$ is the number of markers in $C$ that are also in some bond. Note that in order for (U1.2) to be applied it must be that $S'(D) > 0$. $S'(D)$ can only be increased by application of Step U0 or the SQUEEZE procedure. The total increase over all applications of Step U0 is $O(r)$ by Lemma 7.8. On the other hand, each application of SQUEEZE increases $S'(D)$ by only a small constant. In view of the calculations of the previous paragraph, the total increase over one application of GRAPH is therefore bounded by $O(r)$. But each application of (U1.2) decreases $S'(D)$. It follows that (U1.2) can be applied at most $O(r)$ times.

By the calculations of the previous two paragraphs we may ignore (U1.2), (U1.3), (U2.2) and (U2.3) in counting the number of nodes and the number of members created. The number of applications of (U2.1) is bounded by $O(r)$ because of Theorem 7.5. (U2.4) adds no node or member names (the new member $m(R)$ inherits its name, through (D2) links, from one of the members in $R$). (U1.1) also adds no node or member names. Finally, the number of node and member names created in Step U0 is bounded by $O(r)$ by Lemma 7.8.

●

(7.10) **Lemma**. The total number of edge names created by an application of GRAPH is bounded by $O(r+c)$.

**Proof**. A direct examination of UPDATE shows that no more than $|C-P|+6$ edges can be added by any one application.

●

We now introduce a result of Tarjan [15]. Define the so-called *Ackerman function* $A(i,j)$ on pairs of positive integers $i,j$ by: $A(1,j) = 2^j$, $A(i,1) = A(i-1,2)$ for $i \geq 2$ and $A(i,j) = A(i-1, A(i,j-1))$ for $i,j \geq 2$. For integers $m \geq n \geq 1$, define

$\alpha(m,n)=\min\{i \geq 1: A(i,\lfloor m/n \rfloor) > \log_2 n\}$. The function $\alpha(m,n)$ is *very slow* growing, being for all practical purposes never bigger than 4. ($A(4,1)=f(17)$, where $f(1)=2$ and $f(k)=2^{f(k-1)}$ for $k \geq 2$. Thus, $f(4)=2^{65536}$.)

(7.11) **Theorem.** (Tarjan [14]) The disjoint set union algorithm with path compression runs in time $O(m\alpha(m,n))$ where $m$ is the total number of operations (makesets, links and finds) and $n$ is the number of elements in the underlying set.

●

(7.12) **Theorem.** Given an $r \times c$ $\{0,1\}$-matrix with $n$ nonzero entries, GRAPH runs in time $O(n\alpha(n,r))$ and uses space $O(n)$.

**Proof.** The time bound for Steps G1-G3 follows from (7.1), (7.2), (7.7), (7.9) and (7.11). For Step G4 note that $G$ can be computed directly using (D9) and $O(c)$ (D3) finds. The space bound follows from (7.9), (7.10) and an examination of (D1)-(D11). The determining factor is the storage for the input matrix. We note that some temporary storage is involved in the implementation of several of the steps, principally Steps T1, T3, and H3 (see (6.2)). This storage is not recorded in (D1)-(D11). However, it can easily be verified never to exceed $O(|P| + |\hat{D}|)$.

●

It is open whether there is an $O(n)$ algorithm for graph realization. Recently Gabow and Tarjan [8] have given some conditions under which disjoint sets can be treated in linear time, and although such a reduction in the time required for the applicable parts of GRAPH would imply a linear time bound, our problem does not seem to meet their conditions. Indeed, we conjecture that there is no linear-time (i.e., $O(n)$) algorithm for graph realization.

## APPENDIX. AN EXAMPLE

Consider the 9×8 matrix $M$ in Figure 3:

$$\begin{array}{c|cccccccc}
 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\
\hline
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
2 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
3 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
4 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\
5 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
6 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
7 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
8 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
9 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

**Figure 3:** Input Matrix $M$

$M$ is totally nonseparable since each column, except the first, has a 1 in a row with a 1 in a previous column. We use the labels $B_1, B_2, \ldots$ for the graphs that occur in t-decompositions during the execution of the algorithm.

*Column 10:* In Step G1 and obtain $G_1 = B_1$ with $D_1 = \{B_1\}$ and $pm(B_1) = 10$.

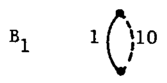

**Figure 4:** $D_1$ (column 10)

*Column 11:* In Step G2 we have $P = \{1\}$. Note that $D = D_1$ is not a t-decomposition since $B_1$ has only two edges, but is a t-decomposition in all other respects.

Step H1 is trivial since $D$ has only one member. In Step H3, $t = 0$, so $H = Q = B_1$. $B_1$ is by convention not a polygon, since it has two edges, and so RELINK2($B_1$) is not applied. Note that $|P| = 1$, and so $P$ is automatically a path. Note also that (R4) and (R5) are applied in Step H3. Since TYPING was not used in this case, (R1)-(R3) were not applied. Thus, $K_1 = K_2 = B_1$. No change occurs in (R5) since $B_1$ is a bond, even though $K_1 = K_2$ and the ends of $P$ are the ends of $pm(B_1) = 10$.

In Step G3 we have $C=\{1,7,8,9,11\}$. The identities of $u_1,u_2$ as required by the input for UPDATE are clear since $B_1$ has only two nodes. In Step U0 we have $C-P=\{7,8,9,11\}$. Let $f=18$. Adding a polygon $B_2$ with edges $\{7,8,9,11,18\}$ to $D^*=\{B_1\}$ we obtain the non-decomposition $D^*$ as in Figure 5 ("non" because there is no marker edge shared by $B_1$ and $B_2$).

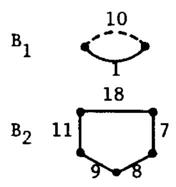

**Figure 5:** $D^*$ for column 11

In Step U1 we apply (U1.1) since, by convention, $B_1$ is not a polygon. This yields the t-decomposition $D_2$ in Figure 6. Note that the marker edge 18 has been oriented. The orientation is arbitrary, as suggested by the discussion following the proof of Lemma 5.1. Note also that edge 18 is given the same name in both $B_1$ and $B_2$. This causes no difficulty here, in contrast to the situation in a computer implementation, as the distinct identities of these two edges is clear from the picture.

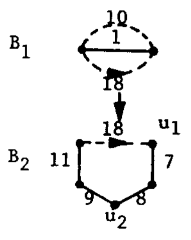

**Figure 6:** $D_2$ (column 11)

*Column 12:* We now have $P=\{7,8\}$. In Step H1 the reduced decomposition is $\hat{D}=\{B_2\}$ since this member of $D$ contains all the edges of $P$. In Step H3, $t=0$, so $H=Q=B_2$. Since $B_2$ is a polygon we apply RELINK2($B_2$). Now $W_Q=P=\{7,8\}$ (see definition of $W_Q$ preceding Lemma 4.3), and $Z=\phi$ since $t=0$ (see definition of $Z$ preceding RELINK1). But $W_Q-Z=\{7,8\}$ is already a path in $B_2$, and so nothing happens. Also no reorientation occurs in Step H3 because $t=0$, or, more generally, because $Z=\phi$. Because of (R4), $K_1=K_2=B_2$. No change occurs in (R5) since $u_1$ and $u_2$ are not adjacent (see Figure 6).

In Step G3 we have $C = \{2,3,4,6,7,8,12\}$, and in Step U0 $C-P = \{2,3,4,6,12\}$. Let $f = 19$. Adding a polygon $B_3$ with edges $\{2,3,4,6,12,19\}$ to $D^*$ we obtain the non-decomposition $D^*$ as in Figure 7.

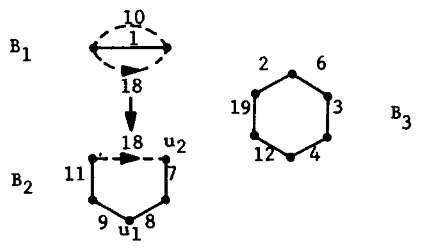

**Figure 7:** $D^*$ for column 12

In Step U1 we apply (U1.3). Letting $f_1 = 20$ and $f_2 = 21$, and where the path $L_1$ has edges $\{9,11,18\}$ and $L_2$ has edges $\{7,8\}$, we obtain the $t$-decomposition $D_3$ in Figure 8.

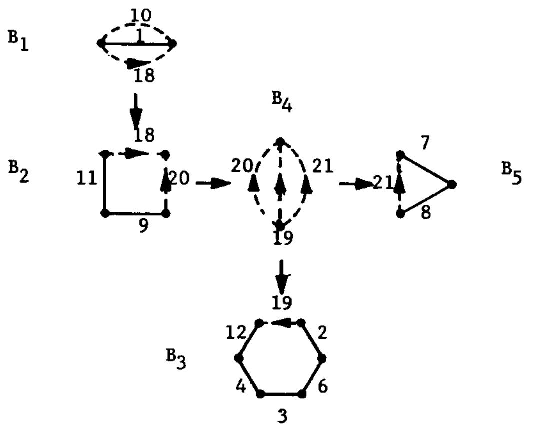

**Figure 8:** $D_3$ (column 12)

*Column 13:* We now have $P = \{2,3,4,6,7\}$. In Step H1 the reduced decomposition is $\hat{D} = \{B_3, B_4, B_5\}$, as given in Figure 9, where the inclusion of $B_4$ is forced by connection. This leads to our first application of TYPING.

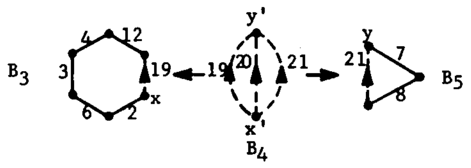

**Figure 9:** $\hat{D}$ for column 13

The depth partition is given by $\pi_0 = \{B_4\}$ and $\pi_1 = \{B_3, B_5\}$, so that $s = 1$. In Step T1 we must apply both RELINK1($B_3$) and RELINK1($B_5$) since $B_3, B_5$ are both polygons. However, in both cases $Z = \phi$; moreover, $W_{B_3} = \{2,3,4,6\}$ and $W_{B_5} = \{7\}$ are both paths incident to the parent markers, 19 and 21, respectively. Hence, no relinking actually occurs, and we have $T(B_3) = T(B_5) = 3$.

We must now apply Step H3. We have $Q = B_4$, $H_1 = B_3$ and $H_2 = B_5$. The current orientation is given by the directions on the marker edges 19 and 21. Since $Q$ is not a polygon we do not apply RELINK2. It remains only to determine a new orientation $f$. As suggested in (4.7), to do this all we need to know about $H_1$ and $H_2$ is that both are type 3, in addition to knowing which parent marker end-nodes meet $W_{H_1}$ and $W_{H_2}$, respectively. These are the nodes $x$ and $y$ in Figure 9. Since their corresponding nodes on $Q$ under $f$, $x'$ and $y'$, are not equal, one of 19,21 must be reoriented. We reorient 19, which yields the reduced decomposition $\hat{D}$ given in Figure 10.

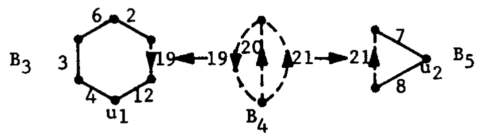

**Figure 10:** Reoriented $\hat{D}$ for column 13

In Step G3 we have $C = \{2,3,4,6,7,13\}$. For the input requirements of UPDATE we note that $K_1, K_2, u_1, u_2$ are chosen in Step T1 by the application of rule (R1). Thus, we have $K_1 = B_3$ and $K_2 = B_5$ ($K_2 = B_3$, $K_1 = B_5$ is an equally valid choice), where $u_1, u_2$ are specified in Figure 10. In Step U0 we have $D^*$ as in Figure 8, except that edge 19 has been reoriented in $B_4$. Since $C - P = \{13\}$ we obtain $f = 13$. Nothing is added to $D^*$ in this step.

Now we come to Step U2. For the path $R$ we have $J_1 = B_3$, $J_2 = B_4$, $J_3 = B_5$, $s = 2$, $m_1 = 19$, and $m_2 = 21$. Note that $B_4$ is a bond, and that $p(B_4) = B_2$ is not in $R$. However, $B_4$ has only three edges. Thus, no change occurs in (U2.1). In (U.2) we apply SQUEEZE($L$) for $L = \{2,3,4,6\}$; all other $L$s encountered have only one edge so that SQUEEZE has no effect in these cases. The resulting $D^*$ is given in Figure 11.

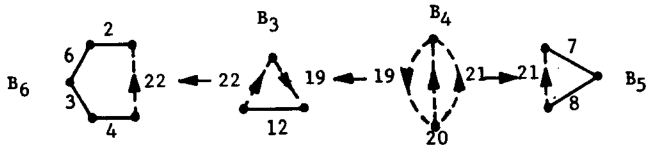

**Figure 11:** $R$ for column 13

Nothing occurs in (U2.3) since $R$ has no internal polygons. Finally, applying (U2.4) we obtain the $t$-decomposition $D_4$ of Figure 12, where $B_3$ is the graph G produced by the merge.

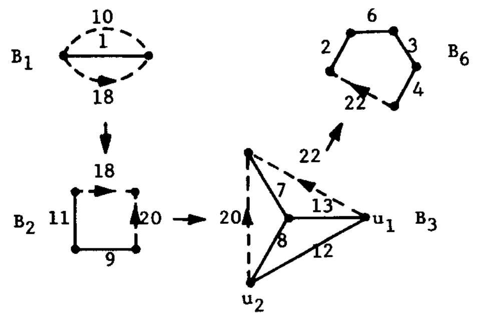

**Figure 12:** $D_4$ (column 13)

*Column 14:* We have $P=\{2,3,4,6,7,8\}$, $\hat{D}=\{B_3,B_6\}$ $\pi_0=\{B_3\}$, and $\pi_1=\{B_6\}$. In Step T1, RELINK($B_6$) has no effect, and $T(B_6)=1$. This brings us to Step H3 with $Q=B_3$, $H_1=B_6$ and $t=1$. Since $T(H_1)=1$ no reorientation occurs. $K_1=K_2=B_3$ because of rule (R4) applied in Step H3, with $u_1,u_2$ as indicated in Figure 12. In Step U0, $f=14$. Finally, (U1.1) is applied yielding $D_5$ as in Figure 13.

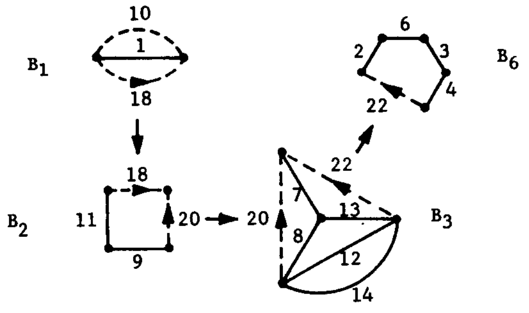

**Figure 13:** $D_5$ (column 14)

*Column 15:* We have $P=\{6\}$ and $\hat{D}=\{B_6\}$. In Step H3, RELINK2($B_6$) switches the positions of 4 and 6, where $W_Q=\{6\}$, $Z=\phi$, so that 6 is adjacent to $pm(B_6)=22$. Because $t=0$, no reorientation occurs, $K_1=K_2=B_6$, by (R4), and $u_1,u_2$ are the ends of 6 in $B_6$. In Step U0 we take $f=23$ (a new element) and introduce a polygon $B_7$ on edge-set $\{f\}\bigcup(C-P)=\{5,15,23\}$. Next (U1.2) is applied, with $f'=6$. Let $f''=24$. Replacing edge 6 by edge 24 in $B_6$, and adding a bond $B_8$ on edges $\{6,23,24\}$ we obtain $D_6$ as in Figure 14.

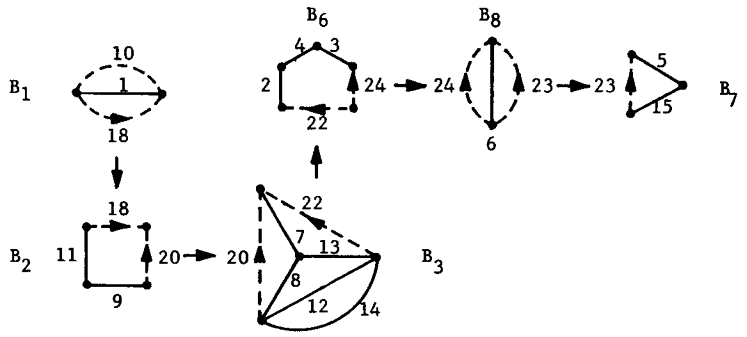

**Figure 14:** $D_6$ (column 15)

*Column 16:* We have $P=\{4,5\}$, $\hat{D}=\{B_6,B_7,B_8\}$, $\pi_0=\{B_6\}$, $\pi_1=\{B_8\}$, and $\pi_2=\{B_7\}$. Applying Step T1 to $B_7$ we see that RELINK1($B_7$) has no effect, and determine that $T(B_7)=3$, $K_1=B_7$ (by rule (R1)), and that $u_1$ is the node of $B_7$ not incident to edge 23. Next we apply Step T3 with $Q=B_8$ and $H_1=B_7$ ($t=1$). No reorientation occurs, and we deduce that $T(Q[H_1])=3$. This completes the application of TYPING, and brings us to Step H3. In applying RELINK2($B_6$), we note that $W_Q=\{4,24\}$ and $Z=\{24\}$, and 4 is switched with 3 to make it adjacent to 24. The application of rule (R4) yields $K_2=B_6$ and $u_2$ equal to the end-node of 4 not shared by 24.

Now in Step U0 we have $C=\{4,5,16\}$ and $f=16$. (U2.1) has no effect. (U2.2) results in squeezing the path $\{2,3,22\}$ from $B_6$. (U2.3) has no effect because $R$ has no internal polygons. Finally, applying (U2.4) we obtain $D_7$ as in Figure 15.

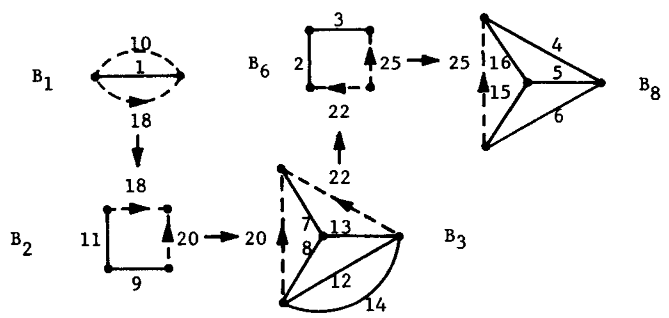

**Figure 15:** $D_7$ (column 16)

*Column 17:* We have $P = \{1,2,3,4,5\}$, $\hat{D} = D_7$, and blocks in the depth partition $\pi_0 = \{B_1\}, \ldots, \pi_4 = \{B_8\}$. In TYPING we obtain $T(H) = 3$ for each application of Steps T1 and T3. RELINK1 is applied twice, once to $B_6$ and once to $B_2$, but in neither case results in any change. The first application of Step T1 results in identifying $K_1$ as $B_8$ and $u_1$ as the node of $B_8$ common to edges 5 and 15. $K_2$ is chosen to $B_2$ in Step H3 since $u_2$ is common to edges 11 and 18. No reorientations occur in TYPING or HYPOPATH. Finally, in UPDATE Step U2 is applied. (U2.2) results in squeezing the path $\{11,9\}$ from $B_2$, and in (U2.3) the path $\{2,3\}$ is squeezed. The application of (U2.4) results in $D_8$ as in Figure 16. $m(D_8)$, a graph realizing the original $M$, is given in Figure 17.

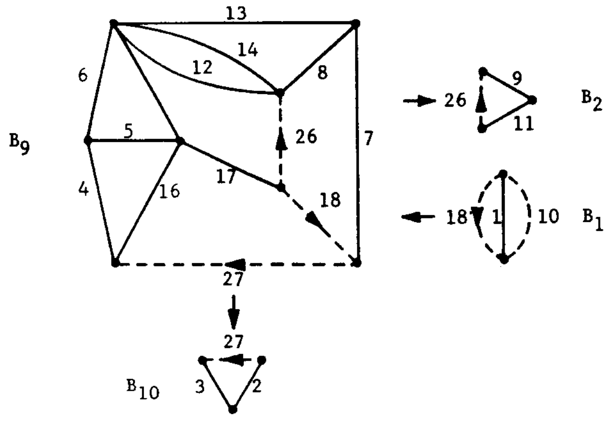

**Figure 16:** $D_8$ (column 17)

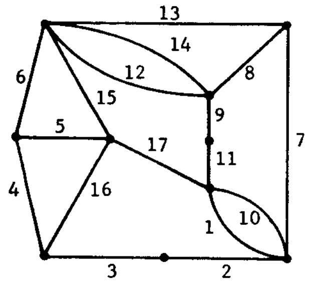

**Figure 17:** A realization for $M$

## REFERENCES

[1] Bixby, R.E. [1984]: Recent algorithms for two versions of graph realization and remarks on applications to linear programming. *Progress in Combinatorial Optimization,* ed. W.R. Pulleyblank, 39-67.

[2] Bixby, R.E. and W.H. Cunningham [1980]: Converting linear programs to network problems. *Mathematics of Operations Research 5,* 321-357.

[3] Bondy, J.A. and U.S.R. Murty [1976]: *Graph Theory with Applications.* North-Holland, New York.

[4] Booth, K.S. and G.S. Lueker [1976]: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms, *J. Comput. System Sci. 13,* 335-379.

[5] Cunningham, W.H. [1982]: Separating cocircuits in binary matroids. *Linear Algebra and its Applications 43,* 69-86.

[6] Cunningham, W.H. and J. Edmonds [1980]: A combinatorial decomposition theory. *Canadian Journal of Mathematics 32,* 734-765.

[7] Fujishige, S. [1980]: An efficient PQ-graph algorithm for solving the graph realization problem. *Journal of Computer and System Science 21,* 63-86.

[8] Gabow, H.N. and R.E. Tarjan [1983]: A linear-time algorithm for a special case of disjoint set union. *Proceedings Fifteenth Annual ACM Symposium on Theory of Computation,* 246-251.

[9] Gavril, F. and R. Tamari [1983]: An algorithm for constructing edge-trees from hypergraphs. *Networks 13,* 377-388.

[10] Hopcroft, J. and R.E. Tarjan [1973]: Dividing a graph into triconnected components. *SIAM Journal of Computing 2,* 135-158.

[11] Iri, M. [1968]: On the synthesis of loop and cutset matrices and related problems. *RAAG Memoirs 4,* 376-410.

[12] Kennington, J.L. and R.V. Helgason [1980]: *Algorithms for Network Programming.* John Wiley, New York.

[13] Löfgren, L. [1959]: Irredundant and redundant boolean branch-networks. *IRE Transactions on Circuit Theory, CT-6, Special Supplement,* 158-175.

[14] Seshu, S. and M.B. Reed [1961]: *Linear Graphs and Electrical Networks.* Addison-Wesley, Reading, MA.

[15] Tarjan, R.E. [1983]: *Data structures and network algorithms.* SIAM, Philadelphia.

[16] Tomizawa, N. [1976]: An $O(m^3)$ algorithm for solving the realization problem on graphs on combinatorial characterization of graphic (0,1)-matrices (in Japanese). *Papers of the Technical Group on Circuit and System Theory of the Institute of Electronics and Communication Engineers of Japan,* OST. 75-106.

[17] Truemper, K. [1980]: On Whitney's 2-isomorphism theorem for graphs. *Journal of Graph Theory 4,* 43-49.

[18] Truemper, K. [1985]: How to detect hidden networks and totally-unimodular subsections of linear programs. *TIMS/ORSA Joint National Meeting,* April, 1983.

[19] Tutte, W.T. [1960]: An algorithm for determining whether a given binary matroid is graphic. *Proceedings of the American Mathematical Society 11,* 905-917.

[20] Tutte, W.T. [1964]: From matrices to graphs. *Canadian Journal of Mathematics 16,* 108-127.

[21] Tutte, W.T. [1968]: Connectivity in Graphs. *University of Toronto Press,* Toronto.

[22] Wagner, D.K. [1983]: An Almost Linear-Time Graph Realization Algorithm. Ph.D. dissertation, Northwestern University.

[23] Wagner, D.K. [1984]: On Theorems of Whitney and Tutte, *Research Memorandum 84-13,* School of Industrial Engineering, Purdue University, West Lafayette, Indiana (to appear in *Discrete Mathematics* ).

[24] Whitney, H. [1933]: 2-Isomorphic graphs. *American Journal of Mathematics 55,* 245-254.